A Novel Autonomous Profiling Method for the Next Generation NFV Orchestrators

Shadi Moazzeni, Pratchaya Jaisudthi, Anderson Bravalheri, Navdeep Uniyal, Xenofon Vasilakos, Reza Nejabati *Senior, IEEE*, and Dimitra Simeonidou, *Fellow, IEEE*

***Abstract*—Currently, telecommunication research communities are striving towards the adoption of Zero-touch network and Service Management (ZSM) in Network Function Virtualisation (NFV) orchestration. Contemporary efforts on adopting Machine Learning (ML) and Artificial Intelligence (AI) have caused an upsurge of ZSM application in the VNF space. While ML and AI complement the ZSM goals for building the intelligent NFV orchestration, a deep knowledge about the resource consumption by Network Services (NSs) and its constituent Virtual Network Functions (VNFs) is required, which would enable AI and ML models to manage the available resources better and enhance user experience. In this paper, we propose a Novel Autonomous Profiling (*NAP*) method that not only predicts the optimum network load a VNF can support but also estimates the required resources in terms of CPU, Memory, and Network, to meet the performance targets and workload by utilising ML techniques. Our performance evaluation results on real datasets show that the output of NAP can be used in the next generation of NFV orchestration.**

***Index Terms*—Profiling, Monitoring, Predictor, Performance KPIs, multi-objective resource configuration.**

I. Introduction

As Network Function Virtualisation (NFV) thrives as a technology that promises the reduction of costs, and the increase in the operational agility in telecommunication service providers' infrastructure [1], the need for Zero-touch network and Service Management (ZSM) techniques becomes more evident.

ZSM pursues the end-to-end automation of network infrastructure life cycle with minimum human intervention through self-configurable, self-healing, self-monitored and self-optimised networks [1]. Although conventional NFV Management and Orchestration systems (MANO) are built to manage Network Services (NSs) in a programmable way, and therefore promote dynamic and efficient resource usage, they are designed to operate upon human action. As exemplified by widespread solutions such as ETSI's OSM [2] and Linux Foundation's ONAP [3], conventional NFV MANO systems only go as far as providing application programming interfaces (APIs) that present very limited autonomy for a few selected scenarios, such as scaling or placement. Even in these scenarios, the provided APIs require extensive input from human operators to work appropriately: deep practical internal knowledge about every particular system need to be encoded in a way that the NFV orchestrator (NFVO) can simply playback a set of pre-established rules.

Authors are with High Performance Networks Research Group in Smart Internet Lab, Department of Electrical & Electronic Engineering, University of Bristol, BS8 1UB, UK, Corresponding author: Sh.Moazzeni (e-mail: shadi.moazzeni@bristol.ac.uk).

To achieve the ZSM goals, the next generation of intelligent NFV MANO systems requires deep information about NSs and their constituent Virtual Network Functions (VNFs) in order to react proactively to the increase or decrease in demands.

Historically the act of acquiring deep knowledge about a computer-centric system is known as Profiling. VNF/NS Profiling systems use monitoring information to create mathematical or computational models for the performance of VNFs and VNF chains, known as profiles. Recently, there have been attempts to use DevOps and ML to extract profiles [4]–[13]. Additionally, various works have studied the integration of these VNF profilers to the existing NFV-MANO architectures [13]; however, the adoption of the profiling solutions into the standard NFV-MANO systems is still awaiting. We understand there are certain limitations for such an integration to be done:

- Firstly, the output of the existing profiling solutions is not designed to directly assist the orchestration and management components of such systems, especially in the life-cycle management of NSs. It either focuses in a particular task (e.g. placement [13]) or is not intended for direct API integration, serving primarily to provide insights to human operators that are in turn, responsible for integrating them into existing systems as custom rules or via other mechanisms.
- Secondly, since existing solutions for NFV MANO do not implement the entire scope of ZSM (instead opting for providing just a few related features), the usage of profiling data requires ad-hoc integration.

The objective of the work reported in this paper is to ease the barriers for the creation of these next-generation intelligent orchestrators. It proposes a novel ZSM-oriented profiling method, called *NAP* (Novel Autonomous Profiling), whose outputs can be directly used for the several tasks in an NFV MANO system.

The paper is organised as follows: state-of-the-art NFV profiling is reviewed in Section II. The role of a profiling system in NFV orchestration is discussed in Section III. Our proposed *NAP* architecture and its associated methods are described in Section IV. The experimentation and evaluation results are presented in Section V, followed by the future works and conclusion in Section VI.

II. Related Works

Classical NFV MANO architectures include monitoring frameworks which allow network operators to check the status of the deployed network services as well as the deployment

platforms. Resource monitoring complements VNF profiling and is required by the profiler to generate the correct VNF profiles. Such monitoring frameworks have been utilised in a staging environment to manage faults in the deployed VNFs and Service Function Chains (SFCs) [14], prior to the deployment on the production environment. These staging environments can provide an insight to the network operator for better utilisation of the available resources while optimising the VNF performance. In this paper, we have created and integrated a custom resource monitoring solution which assists our VNF Profiler to perform autonomous profiling.

Further, in this section, we give an overview of the state of the art NFV profiling related approaches. Table I shows a summary of these works, including their considered resources and metrics, and the methods used to predict the performance measurements and/or the required resources.

Some notable efforts [4]–[13] have been made in the NFV sphere to induce intelligence into the NFV orchestrator while profiling the VNFs. The existing studies can be divided into two parts; Online, for instance, [13], and Offline VNF profiling such as [7], [9], [11]. In addition, the outcome of executing the NFV profiling measurements can be categorised as:

1) Predicting the feasible $metrics$ or performance $KPIs$ under a given configuration of resources.
2) Predicting $configuration\ of\ resources$ for achieving the stated performance KPIs specified in the SLAs. We have called this category as the "Provisioning problem".

Authors in [13] introduce a novel zero-touch orchestration framework called z-Torch, which focuses on the optimal placement of network function depending on the VNF profile. They used unsupervised learning to create the binding affinity of the VNFs (based on the resource requirements), which helps in the decision making for the optimal VNF placement with minimum migration needs. The solution in z-Torch utilises the VNF descriptors to get the KPIs for the VNF and uses some network parameters to estimate the network load for optimal placement. However, we believe, the KPIs provided using VNF descriptors are not optimum and in most cases are over-reserved. Generally, during the deployment, experimenters create the descriptors and define an approximate amount of resources needed by the VNF (mostly, the maximum amount). Whereas, while the NS is running, the consumption of resources by the VNF can change. Also, the assumption of resources can vary from actual consumption. Hence, in our *NAP* architecture, we utilise the VNF descriptors to get the initial resource requirements and run further ML algorithms to find the optimum resource consumption or KPIs for the targeted VNF performance. z-Torch [13] on the one hand is an online VNF profiling tool while on the other hand, is focused on the VNF placement to achieve minimum VNF migration in future. The work presented in this paper focus on the profiling of VNF, considering the network load it can handle for achieving the optimum resource utilisation and is an offline technique based on ML.

Manuel et al. propose an offline profiling system to profile VNFs under realistic resource constraints based on the production environment data [7]. They configure the CPU and measure the application throughput. Subsequently, the same authors introduce a profiling system to profile the entire SFCs [6]. Their SFC-based profiling solution treats the entire SFC as a black box. So, for profiling the whole SFC performance behaviour, it does not consider the profiling results of each constituent VNF individually. The authors consider CPU configuration and measure the overall throughput and response time. However, in both the reviewed works [6], [7], the authors did not utilise ML techniques to improve resource allocation decisions. Similar to this, another VNF modelling approach [4] focus on profiling the VNF as a function of input traffic rate. Authors have utilised Artificial Neural Networks (ANN) to predict the CPU utilisation of a VNF by changing the input traffic rate. Unlike [6], [7], the work done in [4] utilises ML methods for prediction but similar to [6], [7], it does not consider multiple resources (e.g. CPU, Memory, and Network) at the same time .

Manuel et al., in [9] introduce the concept of time-constrained profiling (T-cp). They propose an algorithm to select a limited number of resource configurations under a limited time. They also used regression techniques to predict performance values. Although the authors mention that they can consider more than one resource, in both the Randomised synthetic performance models and also the Real-world performance measurements, each VNF has a single configuration parameter (CPU time). Besides, they have only predicted the maximum achieved throughput as the performance measurement assuming *unlimited* link capacity. Furthermore, to the best of our knowledge, this paper does not provide any clarification on how the profiling technique can be used to tackle the "Provisioning problem". Instead, our *NAP* method has improved the capabilities provided in [9]. On the one hand, our *NAP* method provides algorithms to assign weights to the resources and select the resources that have a higher impact on the performance. On the other hand, various performance metrics and combination of resources are considered simultaneously, including network (Link capacity), to compute the amount of load a VNF can handle. Moreover, by utilising section search techniques, the $optimum$ load it can support has been computed. In addition, our proposed *NAP* method has provided prediction models to address the "Provisioning problem".

Authors in [10] present a componentised method to predict if a given system configuration can process a given load. They produce a prediction label utilising supervised ML technique to show if the load is in line with the CPU capacities. One of the advantages of this work is proposing a component model containing both software and hardware. We believe that our proposed method can be integrated with this approach as our method considers more resources and performance measurements during the model training.

Authors in [15] have profiled the performance of VNFs using various modelling techniques. They provided a resource recommendation algorithm; however, they did not configure the resources simultaneously, and predicted only one performance target. Lastly, the paper [15] lacks a selection algorithm to choose the optimum resource configuration. In comparison, all the three aforementioned shortcomings are addressed in our proposed method.

TABLE I
SUMMARY OF RELATED WORKS; N/A DENOTES "INFORMATION NOT AVAILABLE"

Ref	Considered Resources				Considered KPIs/Metrics				ML Techniques used	Predicted Metrics	Provisioning problem Predicted Resources			
	CPU	Memory	Network	Comments	CPU_Util	MEM_Util	Latency	Comments			CPU	Mem	Net	Comments
z-Torch [13]	✔	✔	✔	Online Resource-Profiling	✔	✔	✔	VNF Profile deviation-is monitored	Q-Learning, Unsupervised ML	Binding Affinity				optimal VNF-placement
[4]	✔			VNF as a function of-traffic	✔				Artificial NN	CPU Consumption	✔			
Offline profiling etc. [7]	✔							Throughput	N/A	N/A				N/A
SFC profiling etc. [6]	✔							Throughput and Response time	N/A	N/A				N/A
WRVS2 etc. [9]	✔							N/A	SVRPRK, DTRP, LRP, RRP	Throughput				N/A
componentised approach [10]	✔			CPU & Server shared resources,VNI,CP,VL			✔*	Packet loss, *Delay	Supervised ML	Load				N/A
Profile-based etc. [15]	✔		✔*	*Bandwidth allocation-only for one VNF	✔			(Packetsize,# of flows) or (file size,cache hit ratio)	Regression, kNN, Interpolation, ANN, Curve Fit	Packet Rate or Response time	✔			
Proposed Method (*NAP*)	✔	✔	✔		✔	✔	✔		MIMO GRNN, RF, and MLP	MIR	✔	✔	✔	

Predicting more than one performance measurement is investigated in [11] and [16]. Authors in [16] suggest the multiple-input–multiple-output (MIMO) model using a general regression neural network (GRNN) to simultaneously estimate the amount of five traffic-related environment pollutants (SOx, NOx, NH3, NMVOC, and PM10). In addition, the accuracy of the model was enhanced by considering various error metrics. Thus, this approach encouraged us to utilise this MIMO method besides the other ML techniques in profiling the VNFs. On the other hand, [11] presents queuing-based modelling for SFC as well as a method to perform automatic scaling by considering only the overloaded individual VNFs. In their proposed method, metrics including the service rates and packet arrival rates are considered to predict performance measurements such as latency, throughput, and response time. The use of queuing theory seems to be an auspicious method to be utilised in our future work. However, the configuration of resources such as CPU, Memory, and Link capacity is not tackled in [11], which is very critical for profiling and is addressed in our proposed method.

The majority of the existing approaches provide methods to predict performance metrics under given resources (first category introduced at the beginning of this section). In contrast, less work focus on the "provisioning problem" to predict the required resources to achieve the given KPIs (second category). However, as can be seen in Table I, none of them targets the two categories simultaneously. Furthermore, the numerical results in Section V show that based on the type of VNF under profiling, some resources have a more significant impact on increasing or decreasing the performance KPIs comparing with the other resources. So, it is important to assign weights to the resources based on their impact on the performance KPIs. Then, given the limitation of profiling time and the fact that all possible configuration of resources cannot be selected and tested, the profiler should select only the resources that have a greater impact on the KPIs (with a higher weight) for that VNF type. Our proposed *NAP* method targets both the categories utilising several ML techniques and considers the weight of various resources such as CPU, Memory, and Network in profiling the given VNF in a limited profiling time. In addition, it predicts the optimum network load that a VNF can support while achieving all the KPIs.

III. THE ROLE OF PROFILING IN NFV ORCHESTRATION

When deploying an NS in the form of an SFC composed by multiple VNFs, various resources like virtual CPU cores, Memory, and NIC should be assigned to the involved VNFs to meet different performance targets [17], or even fulfil Service Level Agreements (SLAs). Moreover, the orchestrator must ensure that virtual links are established accordingly to support the demands in these VNFs, which determines the underlying network requirements (such as bandwidth, maximum delay).

Therefore, the main role of a profiling system is to uncover this relationship between the resource configuration, service demand and performance targets. The conventional approach for NFV profilers is to expose the VNFs or NSs to a varying traffic source while recording resource usage in order to obtain a model that correlates the service demand and resource usage. This approach is derived from the assumption that the maximum service demand supported (D) is a function (f) of the resource configuration (R), and optionally the performance goals (P), $D = f(R, P)$. Thus, the generated profile (f') is a model that approximates this function, $f'(R, P) \approx f(R, P)$. It is important to observe that the profile does not necessarily correspond to a mathematical function, sometimes it can be given by a computer program, simulation, parameters of an ML technique or even by something as simple as a table.

Alternatively, a slightly different approach might attempt to approximate a function (g) that correlates the observed performance (P_o) with the resource configuration and the observed service demand (D_o), $P_o = g(R, D_o)$. While both attempts are valuable (e.g. to assess a specialist during the design of the VNF graph and its scaling rules or a business unit during the evaluation of SLAs), they are not able to directly provide values for resource configuration to the NFV MANO system. Instead, an additional convoluted technique must be applied (such as search algorithms, reinforcement learning, or cognitive techniques).

As indicated by Figure 1, a simple analogy between classical control systems [18] and an intelligent NFV MANO system evidence that, in order to be directly used, the obtained profile (h') should compute the resource configuration based on the performance goals and the monitored metrics, $R = h(D, P)$, $h'(D, P) \approx h(D, P)$; similarly to the classical controller that calculates the actuation input based on a reference and the values measured by a sensor. This proposed approach is used

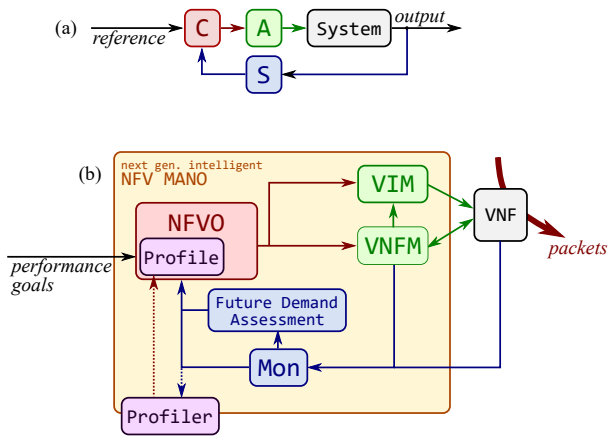


Fig. 1: Analogy between **(a)** classical control loops containing a controller (C), actuators (A) and sensors (S), and **(b)** next generation intelligent NFV MANO systems, that feed metrics acquired by a monitoring system (Mon) into a model (Profile) to calculate the appropriate resource configuration.

in the following sections as the basis of the design of an advanced NFV Profiler.

The information contained in the profiles of the different VNFs composing an NS are fundamentally required by an intelligent MANO system in two distinct phases.

During **day 1** deployment, the NFVO uses these models to estimate the resources needed for the NS instantiation based on the expected demand for the service. This resulting resource configuration can then be considered as input for VNF placement algorithms and topology optimisations.

During **day 2** operations, the NFVO will use the same models to perform common NS life-cycle management tasks. In other to do that, the NFVO continuously receives data from the existing monitoring tools – ideally aided by a series of algorithms that assess the demand for the service in the near future – and based on this data identifies the main VNFs responsible for eventual performance degradation. After the identification, the NFVO will act together with the VNF Manager (VNFM) to eliminate the existing (or potential) bottlenecks via horizontal and vertical scaling. In this phase, the profile is again used to estimate the resource configuration that is more appropriate for the new demand target. Migration is also considered when scaling VNFs using the current Network Functions Virtualisation Infrastructure (NFVI) nodes or networks result in sub-optimal performance. Finally, the Virtual Infrastructure Manager (VIM) interfaces are triggered to enforce the new resource configuration.

Notice that in this context, profiling can be either done completely offline and simply loaded into the NFV MANO system or integrated into it. While offline profiling simplifies the complete implementation, integrated - or online - profilers can dynamically revise existing models based on up-to-date metrics coming from the monitoring system, which in theory, should improve accuracy.

It is important to distinguish between **reactive** and **proactive** usages of such profiles in day 2 operations. **Reactive profile-based MANO** can be understood in the light of the control system analogy (as indicated in Figure 1). As the delay related to training the models is negligible, once the VNF performance degrades, the NFVO shall use the information in the profile ("Performance dataset" or utlise the Predictor manager) to re-configure the VNF (or more generally, the VNF chain) as an attempt to recover its previous performance characteristics. The same reasoning can be applied for reducing the allocation of resources upon decreasing demand. On the other hand, **proactive profile-based MANO** is achieved once the NFVO has some form of assessment about the future demand (or input rates), e.g., based on hand-written policies or another separate mathematical/ML model that uses historical data. With this kind of information, the NFVO can change the configurations before observing any performance impairment in order to be more resilient. For instance, by utilising our predictor manager, it can predict the absolute configuration of resources that is required for meeting with the given KPIs and Optimum MIR in the target environment, and run the VNF with the proposed configuration of resources, proactively.

TABLE II
NOTATIONS & DEFINITIONS

Parameter	Definition
Profiling Workflow:	
t	The profiling timer.
$Pref\_Conf$	The Preferred configuration of resources allocated to the *VNF*.
$Pref\_Conf_R$	The Preferred configuration of a res. (R) allocated to the *VNF*.
W_R	The weight of resource (R).
MIR_{avg}	Opt. MIR the *VNF* can handle with average conf. of resources.
MIR_{R}	Opt. MIR the *VNF* can handle with maximum conf. of the resource R.
Predictor Models and Training:	
D	Euclidean distance matrix of dimension $p \times m$.
Evaluation metrics:	
SS_{RES}	The residual sum of squared errors of the regression model.
SS_{TOT}	The total sum of squared errors.
y_i	The actual or the observed value for the experimental unit i.
$\hat{y}_i$	The predicted or the fitted value for the experimental unit i.
$\bar{y}$	The mean of y value.

IV. Profiler Design

To fulfil the vision presented in Sec. III while addressing the knowledge gaps previously indicated, our proposed *NAP* method assimilates several characteristics from related works (Sec. II), while implementing a new set of algorithms and procedures. For the sake of simplicity, an offline design is chosen. Nonetheless, the proposed method can be integrated into an orchestrator to perform real-time updates on the models obtained offline, hence achieving *dynamic profiling*.

Although some works indicate that profiling an entire VNF chain is more precise than profiling its individual VNFs and combining the results [6], our implementation can distinguish every atomic part of a VNF chain, identifying their contribution to the overall NS. This is a fundamental requirement for

NFV MANO, as it is a common practice to promote localised changes in graph topology of the VNF chain during runtime to respond to demand variations or quality of service degradation, (e.g., horizontal scaling of bottlenecks, and migration of sub-parts of the chain). Furthermore, profiling individual VNFs imposes *less* resource requirements for the profiler itself: focused, small tests are simpler, quicker, and cheaper to run. Also, individual profiles favour composition: not only a service provider can use individual profiles to compare and choose between VNFs that execute the same functionality, but also by re-using already profiled VNFs to compose new chains, new profiling rounds can be completely avoided. Therefore, our design focuses on single VNF profiling.

Ideally, each VNF should be profiled in the whole range of possible input workload values and all available resource configurations [7]. However, this leads to an ample space of multiple parameters and many combinations to test, resulting in an expensive and overextended measurement period [9]. On the other side, given the huge resource configuration space of a VNF, and the fact that its re-deployment or re-configuration takes a considerable amount of time, executing profiling measurements over the complete configuration space is infeasible [9]. To overcome this issue, we select and profile only a small subset of all possible resource configurations. By design, the Predictor Manager component of our proposed *NAP* method creates and trains prediction models. These models are capable of predicting certain quantities after past measurements by utilising regression techniques or more complex ML solutions such as MIMO ANN models. The first role of these models is to accurately predict the *Optimum maximum* input rate ($Optimum\ MIR$) for previously *untested* resource configurations which meet the given KPIs. Their second role is to calculate the absolute amount of resources required for meeting both the given KPIs and the $Optimum\ MIR$ in the target environment. Both roles can be achieved in parallel by utilising the ML-based techniques described in Sections IV-D and V-E.

A. Profiling Platform

Figure 2 shows the high-level architecture of our *NAP* method, including its main building blocks. This architecture assumes that each VNF is associated with a list of performance targets (or even SLAs) and relevant profiling parameters. The VNF descriptor (VNFD) refers to the VNF catalogue, a set of templates that describe the deployment and operational characteristics of available VNFs as well as their images. As per this, the VNFDs could also include resource requirements such as CPU, RAM, and Disk space. Therefore, the VNF Descriptor and the Profiling Parameters specify which resources should be tested throughout the profiling process, the **minimum** and **maximum** values per resource, and the performance metrics to be collected and analysed. The performance targets are evaluated via KPIs that decide if the VNFs are successfully handling the workload or not. The ***Profiler*** module resides as part of the Profiling platform and has three main building blocks: the ***Weighted Resource Configuration Selector***; the ***Analyser and Post Processor***; and the ***Predictor Manager***.

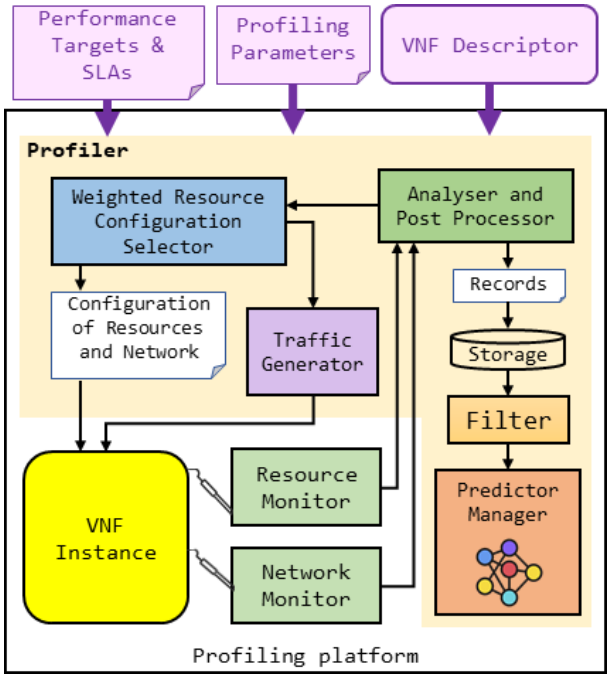


Fig. 2: High-Level architecture of our proposed *NAP* method including the main building blocks

The ***Weighted Resource Configuration Selector*** gets the VNF descriptor, profiling parameters and the list of performance targets. It selects a set of resource configurations and assigns them to the VNF, and then runs it with these allocated resources. Afterwards, it requests for traffic to be generated and sent by the *Traffic Generator* to the *VNF instance*. Alongside the help of the ***Analyser and Post Processor***, it queries the monitoring data from the monitoring tools (*Resource Monitor* and *Network Monitor*) to find and record the $Optimum\ Maximum\ Input\ Rate\ (Optimum\ MIR)$ that the *VNF instance* can handle to meet all performance targets. The outcome of this step is referred to as the "VNF Performance Dataset".

The ***Predictor Manager*** is designed to create and train prediction models that are capable of predicting certain quantities, based on past measurements. It has two roles: 1- it creates a model to accurately predict the $Optimum\ MIR$ for previously *untested* resource configurations in which the given KPIs can be met; 2- it calculates the absolute amounts of resources required to meet both the given KPIs and the $Optimum\ MIR$ in the target environment. Both above can be achieved simultaneously by utilising several ML-based techniques. These offline techniques can meet with the needs of fairly static network conditions and, hence, be deployed to such environments with good profiling performance expectations. Likewise, performance expectations can be also met for short term period deployments. These trained models can be used to bootstrap profiling, which can be further continued with other models such as online trained ones. Finally, the models that have been used in this work are in practice faster regarding their response times as well as more accurate under stable conditions [19], [20]. These models can then be used by the next generation intelligent NFV MANO (see Figure

1) to optimise their resource dimensioning decisions. The weighted selection of the resource configurations as well as the process of computing the $Optimum\ MIR$ are described next in Algorithm 1 and in Figure 3.

B. Profiling Workflow

The profiling workflow as shown in Algorithm 1 comprises five steps; The notations and definitions used are shown in Table II and corresponding details are provided in the following. In step (1), the profiler sets the **minimum** configuration for all the resources ($Pref\_Conf_R \leftarrow$ Min) and runs the VNF with these allocated resources by calling Run_VNF Function. $Pref\_Conf$ is an array which shows the preferred configuration of resources allocated to the $VNF\ instance$. For example, if the VNF can have 1-16 CPU cores, 128-2048 MB Memory, and 128-4096 Mbps Link capacity, the **minimum** configuration for CPU, Memory, and Link capacity is equal to 1, 128, and 128, respectively. So, $Pref\_Conf$ for CPU, Memory, and Link capacity will be 1, 128, and 128, respectively (i.e., $Pref\_Conf_{CPU} \leftarrow 1$, $Pref\_Conf_{Memory} \leftarrow 128$, and $Pref\_Conf_{Link\ capacity} \leftarrow 128$). Then, the Algorithm shown in Figure 3 will be used to find the $Optimum\ MIR$ the $VNF\ instance$ can handle with **minimum** configuration of resources while meeting the performance targets.

In a similar way, step (2) sets the **maximum** configuration for all the resources. Based on the example given, it will be $Pref\_Conf_{CPU} \leftarrow 16$, $Pref\_Conf_{Memory} \leftarrow 2048$, and $Pref\_Conf_{Link\ capacity} \leftarrow 4096$. Then it will run the VNF and will find the $Optimum\ MIR$ for this configuration of resources.

In step (3), the profiler allocates the **average** configuration of resources to the VNF and will run it. Following our example, it will be equal to $Pref\_Conf_{CPU} \leftarrow 8$, $Pref\_Conf_{Memory} \leftarrow 1088$, and $Pref\_Conf_{Link\ capacity} \leftarrow 2112$. Then, it will find the $Optimum\ MIR$ that the VNF can handle with **average** configuration of resources; We refer to this traffic rate as the MIR_{Avg}.

In step (4), the profiler computes the weight of each resource (W_R) by checking its impact on the $Optimum\ MIR$. To check the impact of each resource (R), it selects the **maximum** configuration of that resource and the **average** for other resources. Then, it will run the VNF with these allocated resources and will find the $Optimum\ MIR$ that the $VNF\ instance$ can handle. We refer to this traffic rate as the MIR_{R}. Afterwards, the profiler will compute the weight of that resource (W_R) as defined in formula (1). For instance, based on the previous example, to find the MIR_{CPU} and then the W_{CPU}, the VNF will be executed with 16 CPU cores, 1088 MB Memory, and 2112 Mbps Link capacity. It is worth mentioning that before computing the weights, the related input rates were normalised by utilising feature scaling, and the process of finding each weight is repeated ten times to improve the accuracy of the weights.

$$W_R = abs(MIR_R - MIR_{Avg})/MIR_{Avg} \quad (1)$$

Finally, through step (5), the selection of resources will be prioritized by considering their weights (W). According to the computed weights of resources, each resource may have a different probability for being selected. So, the profiler randomly selects a resource R from the list of $Resources$ according to the relative sequence of their weights. Then, it will select a random configuration (i.e., a random number between the **minimum** and **maximum** amount) for that resource, R. It is noted that all the other resources will be configured as **average**. Afterwards, the $Optimum\ MIR$ for this configura-

Algorithm 1: Weighted Resource Configuration Selection (WRCS)

Input: VNF, Profiling parameters, Performance targets
Output: Datasets, Weight of each resource (W_R)

Function WRCS(***Input***):
  t=0; // t is the profiling timer
  Profiling step(*1*):
    foreach R *in Resources* **do**
      $Pref\_Conf_R \leftarrow$ Min;
    end
    Call run_VNF($Pref\_Conf$);
    Find $Optimum\ MIR$ through Figure 3;
  Profiling step(*2*):
    foreach R *in Resources* **do**
      $Pref\_ConfR \leftarrow$ Max;
    end
    Call run_VNF($Pref\_Conf$);
    Find $Optimum\ MIR$ through Figure 3;
  Profiling step(*3*):
    foreach R *in Resources* **do**
      $Pref\_Conf_R \leftarrow$ Avg;
    end
    Call run_VNF($Pref\_Conf$);
    $MIR_{Avg} \leftarrow$ Find $Optimum\ MIR$ through Figure 3;
  Profiling step(*4*):
    foreach R *in Resources* **do**
      $Pref\_Conf_R \leftarrow$ Max;
      $Pref\_Conf_{otherResources} \leftarrow$ Avg;
      Call run_VNF($Pref\_Conf$);
      $MIR_R \leftarrow$ Find $Optimum\ MIR$ through Figure 3;
      $W_R = abs(MIR_R - MIR_{Avg})/\ MIR_{Avg}$;
    end
  Profiling step(*5*):
    while $t < Profiling\ time$ **do**
      $R \leftarrow$ random choice ($Resources$,W);
      $Pref\_Conf_R \leftarrow$ Pick a random configuration for R;
      $Pref\_Conf_{otherResources} \leftarrow$ Avg;
      Call run_VNF($Pref\_Conf$);
      Find $Optimum\ MIR$ through Figure 3;
    end
End Function
Function run_VNF($Pref\_Conf$):
  Run VNF with $Pref\_Conf$ and set the networks;
End Function

tion will be computed, and the respective record will be stored. For instance, if W_{CPU} is greater than the other resources, the CPU has a higher probability to be selected. Assuming the profiler has selected the CPU, following our example, the selected configuration will be equal to $Pref\_Conf_{CPU} \leftarrow$ a random number between 1-16, $Pref\_Conf_{Memory} \leftarrow$ 1088, and $Pref\_Conf_{Link\ capacity} \leftarrow$ 2112. The profiler will continue this randomly weighted selection of resources and will add the records to the "VNF Performance Dataset" until the end of profiling time. This profiling time is specified in the Profiling Parameters.

C. Method to Find Optimum MIR

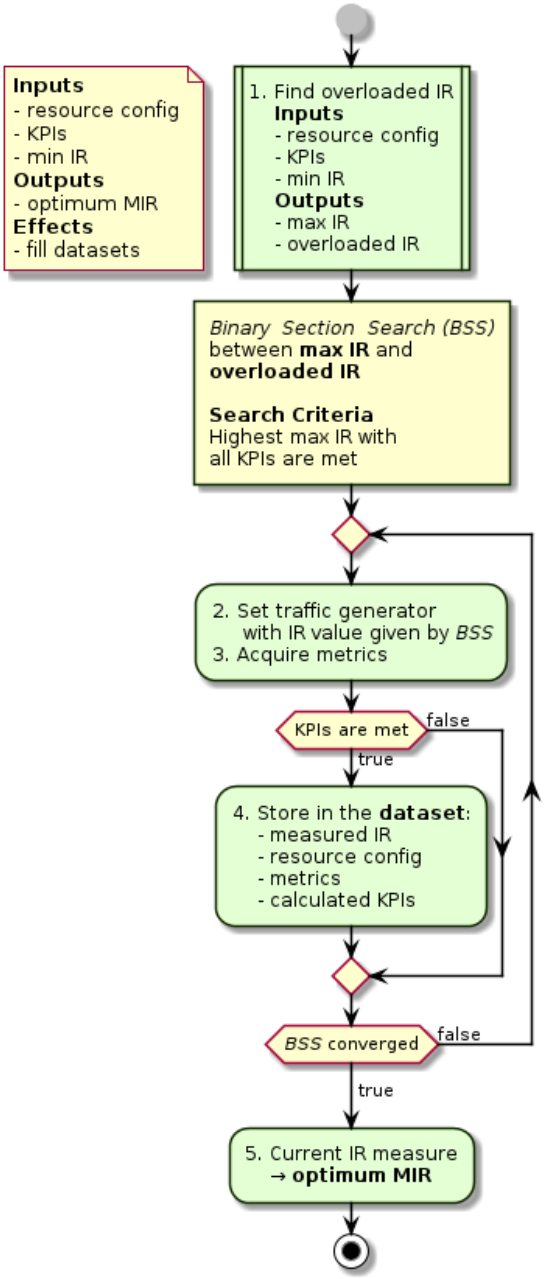


Fig. 3: Algorithm for Finding the $Optimum\ MIR$. Internally it calls the Algorithm in Figure 4

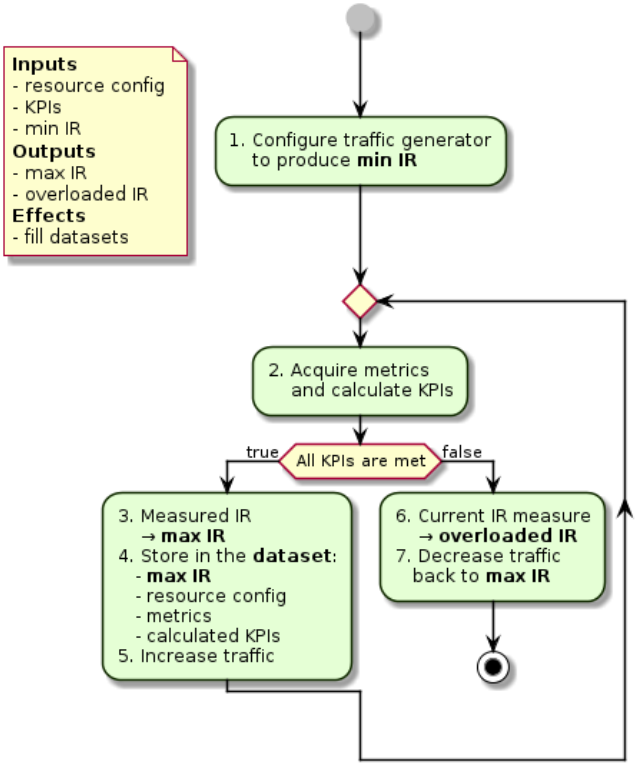


Fig. 4: Algorithm for finding the Overloaded IR

As described in previous subsections, the $Optimum\ MIR$ is considered as a performance metric. It shows the maximum load in terms of the traffic rate a *VNF instance* with an allocated configuration of resources can handle while meeting the performance targets. To compute this metric, the $Find\ Overloaded\ IR$ function will be utilised (illustrated in Figure 4).

- First, the profiler (through the ***Weighted Resource Configuration Selector***) sets the $Traffic\_Rate$ to a given minimum IR and requests the *Traffic Generator* to send UDP packets to the *VNF instance* with the selected rate. In our profiling method, the *Traffic Generator* component is a "pluggable" component; for the time being, the implementation is based on iPerf3, which can generate UDP and TCP packets. So, in case injecting the TCP packets is needed, we can generate TCP packets and monitor and profile the performance of VNFs. In the experiments, UDP was preferred since it results in more precise throughput measurements.
- Next, the profiler (through the ***Analyser and Post Processor***) retrieves the monitoring data and computes the performance metrics such as CPU utilisation, memory utilisation and latency.
  - While the measured performance metrics meet the required KPIs; it will record the $Traffic\_Rate$, $Pref\_Conf$, the metrics captured from the Monitoring tools and the measured performance corresponding to the required KPIs. In addition, it will feedback the ***Weighted Resource Configuration Selector*** to increase the amount of $Traffic\_rate$ and send the

increased amount to the *VNF instance*.
  - When any of the measured performance metrics stops meeting the performance targets, the $Find\ Overloaded$ function will return two traffic rates; The traffic rate that could be handled by the system referred to as the "$Max\ IR$" and the increased traffic that resulted into not meeting the KPIs, called "$Overloaded\ IR$". It is assumed $Min\ IR$ can always be handled.
- Then, returning to Algorithm shown in Figure 3, a section search function such as BSS or Golden Section Search (GSS) method [21] will be called recursively to find and return the $Optimum\ MIR$ for the *VNF instance* with the allocated configuration of resources while allowing all the given KPIs to meet at the same time.
- Finally, the profiler (through the ***Analyser and Post Processor***) will add this record to the "VNF Performance Dataset".

D. Predictor Models and Training

We investigate three ML alternative techniques employed by the ***Predictor Manager*** to generate the predictor-based profiles: Multiple-Input-Multiple-Output General Regression Neural Networks (MIMO-GRNNs), Random Forest (RF), and Multi-Layer Perceptron (MLP). The corresponding details are provided below.

1) MIMO-GRNN: A MIMO-GRNN is a type of a feed-forward neural network that consists of four layers [16]. The *input layer* calculates the Euclidean distance between trained records and test records. The second layer, i.e. the *pattern layer*, produces the exponential activation function. The third layer (*summation layer*) produces a weighted sum matrix and an unweighted sum matrix. The cross product of the above matrices results in multiple outputs. Finally, there is a *output layer* for the model prediction.

2) Random Forest: RF is a model encompassing many decision trees trained together [22]. Each tree receives features randomly to produce separate predictions by itself. The most voted prediction is chosen as the final result, a technique that is known as *Bagging*. Important parameters to tune for an RF include the number of features and the number of trees.

3) Multi-Layer Perceptron: An MLP [23], is a type of ANN, a feed-forward network that is typically used for addressing non-linear data. In such a type of network, the number of hidden layers can be specified following the dataset and the requirements. To avoid both *overfitting* and *underfitting*, an appropriate setup must be considered regarding the learning rate, the optimiser, and the activation function for training profiled datasets.

V. Implementation, Data analysis and Evaluation

This section provides a detailed description of the experimental setup, the implementation of our proposed *NAP* method, and how to generate and analyse adequate "VNF Performance Datasets".

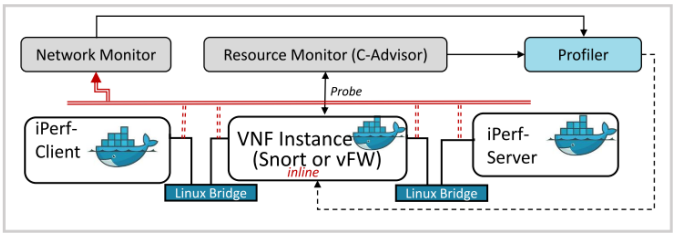


Fig. 5: Experiment Setup

A. Experimental setup

In our experiments, the profiling platform was deployed on a dedicated bare-metal server with a 24-core Intel(R) Xeon(R) CPU running at 2.40GHz, with 160 GB of RAM. Further to that, the software specifics are summarised in Table III. In our profiling platform, we have profiled two types of VNFs; an Intrusion Detection System (IDS) - SNORT [24] and a virtual Firewall (vFW) following the setup shown in Figure 5. vFW acts as a basic packet filtering to allow or block network traffic on specified TCP/UDP port numbers. SNORT uses several detection rules combining signature, protocol, and anomaly rules to detect and act on any malicious activity. Besides, SNORT is used in the inline mode where the two interfaces (depicted in Figure 5) are bridged together to forward the packets from the IPerf3 client to the IPerf3 server and vice-versa. It is important to note that our profiling method is capable of profiling any VNF (black-box approach), not only the two mentioned.

To profile each VNF, several resources, including *CPU*, *Memory*, and *Link Capacity* are tested to measure performance KPIs. The latter KPIs include the $Optimum\ MIR$ and Latency, and metrics such as CPU utilisation and Memory utilisation. As shown in Figure 5, the network latency in terms of maximum Round Trip Time (RTT) is monitored and logged in the Network Monitor, which feeds the network monitoring data to the Profiler. Similarly, the Resource Monitor monitors the compute resource utilisation by probing into the *VNF instance*. Since all the components in the network are hosted as docker containers, we used cAdvisor [25] as a Resource Monitor wrapped around with the custom python APIs to interface with the Profiler.

Based on Algorithm 1 (Section IV) we profiled each VNF with the following resource configuration: (a) CPU between 0.3 and 1.0 vCPU cores, (b) memory between 33 and 100 MB, (c) link capacity between 350 and 650 Mbps. For each resource, the first number shows the **minimum** and the latter

TABLE III
Software used for evaluation experiments

Software	Function	Version
Docker	To run VNFs as containers	19.03.5
SNORT	Intrusion Detection System (IDS)	2.9.15.1
vFW	virtual Firewall docker container	1.6.1
iPerf3	Tool to measure the network	3.1.3
Linux	The OS on each machine	Ubuntu18.04
cAdvisor	The resource monitoring tool	Latest
Elasticsearch	data storage, search, and analytics engine	7.6.0
Kibana	Tool to visualise the Elasticsearch data	7.6.0

one refers to the **maximum** configuration. The considered performance metrics and KPIs were: (i) CPU utilisation $\geqslant$ 97$\pm$3%, (ii) memory utilisation $\leqslant$ 98%, (iii) latency $\leqslant$ 2.5$\pm$0.5 ms, (iv) IR $\geqslant$ 100 Mbps, and (v) total profiling time $\leqslant$ 48h.

B. Generating adequate VNF Performance Datasets

The profiler sets the timer and runs the VNF under profiling (SNORT or vFW) with **minimum** configuration of resources. Then, by sending the traffic(starting from the $Min\ IR$) through iPerf3, it finds the $Optimum\ MIR$ that the *VNF instance* can handle to meet the performance Metrics and KPIs' threshold. Afterwards, it creates a record from the results. The profiling continues by finding the $Optimum\ MIR$ for the **maximum** and later with the **average** configuration of resources and the corresponding records are created. It is important to mention that the average time to run the docker container is around 1.5 second and the average time taken to select the resources utilising binary search for one record of performance profiles is less than 1.6 minutes. The average values of obtained $Optimum\ MIR$s related to the **maximum**, **minimum**, and **average** configuration of resources are shown in Figure 6, with filters F1, F2, and F3, respectively. Following this, the profiler uses Algorithm 1 to compute the weight of each resource and then records the values. Filters F4, F5, and F6 in Figure 6 show the average values of $Optimum\ MIR$ for computing the weights of CPU, Link capacity, and Memory, respectively. As can be seen in Figure 6, the link capacity has the greatest impact on the $Optimum\ MIR$ for vFW than the other resources. So, it should have a greater weight when profiling the vFW. In comparison, both CPU and link capacity significantly affect the $Optimum\ MIR$, and their weights are also higher than the other resource, i.e., memory. The profiler continues to profile the *VNF instance* by the weighted random selection of resource configurations until the end of profiling time. In addition, it stores the generated "VNF Performance Dataset" records utilising the *Elasticsearch*, Logstash, and Kibana (the Elastic Stack) [26] data repository. Employing the Elastic stack gives developers the advantage of a 'store first, examine later' philosophy of monitored measurements. In addition, this dataset can serve the purpose of future analysis by the next generation of intelligent NFV MANO systems, while can also be utilised by the prediction models.

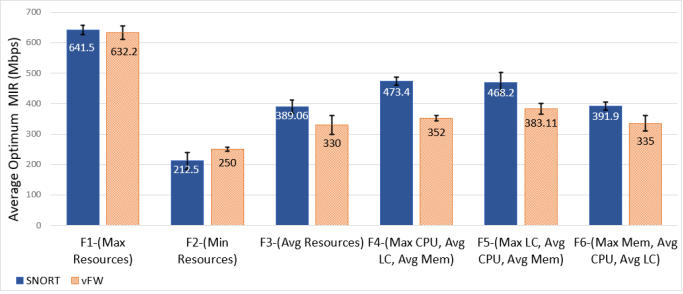


Fig. 6: Average $Optimum\ MIR$ annotated with 95% confidence intervals obtained to compute the weight of resources for SNORT and vFW.

C. Analysing VNF Performance Datasets

To analyse the effect of various resources such as CPU (number of VCPU cores), $Network$ (Link capacity in Mbps), and $Memory$ (in MB) on the $Optimum\ MIR$ (in the rate of megabits per second) the "VNF Performance Dataset" are explored from the Elastic stack repository and illustrated in Figure7 a, b, and c, respectively. Noted that for analysing the effect of each resource, the other resources are filtered as average. Following Figure 7, SNORT VNF, a clear correlation between $Optimum\ MIR$ can be seen increasing with CPU (a) and $Link\ Capacity$ (b). This can be interpreted as the higher the CPU and $Link\ Capacity$, the greater the load the SNORT can handle, while $Memory$ seems to have the least correlation on the Optimum MIR. In comparison, when analysing the performance profiles of vFW, the $Link\ Capacity$ (b) has the greatest effect on the $Optimum\ MIR$, while $Memory$ and CPU seem to have the least correlation on the $Optimum\ MIR$. As the profiling time is limited, these results show the accuracy of our model and the necessity of computing the weights of various resources to profile a system for improved overall performance.

D. Evaluation metrics

In what follows, we define the error-based evaluation metrics used for comparing the models we introduced in Section IV-D. In brief, we employ Mean Average Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared values. On the one hand, MAPE, MAE and RMSE are used to conclude on the best model approach with respect to datasets. The smaller the value of these error metrics, the higher the accuracy of prediction. R-squared, on the other hand, is used to assess how much the regression line fits with data, and to compare the fitted regression line to a mean line. Both groups of error metrics serve important roles for the Prediction Manager in NAP, by adopting increased accuracy in prediction while assessing the trend for overfitting or underfitting to training data. The notations and definitions used in these evaluation metrics are displayed in Table II.

MAE defined in formula (2) is a simple easy to interpret and is insensitive to outliers' metric:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{2}$$

This metric is ideal for easy and fast NAP results.

RMSE defined in formula (3) is appropriate for the regression problems that can be faced by the NAP's Prediction Manager. Its value is always greater or equal to MAE. RMSE considers an unequal weighting scheme with large errors being given a higher weight. However, outliers have an effect on RMSE, thus they should be removed before training the ML model. Evidently, RMSE should be used alongside MAE or other metrics by the Prediction Manager for VNFs with outlying input traffic.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{3}$$

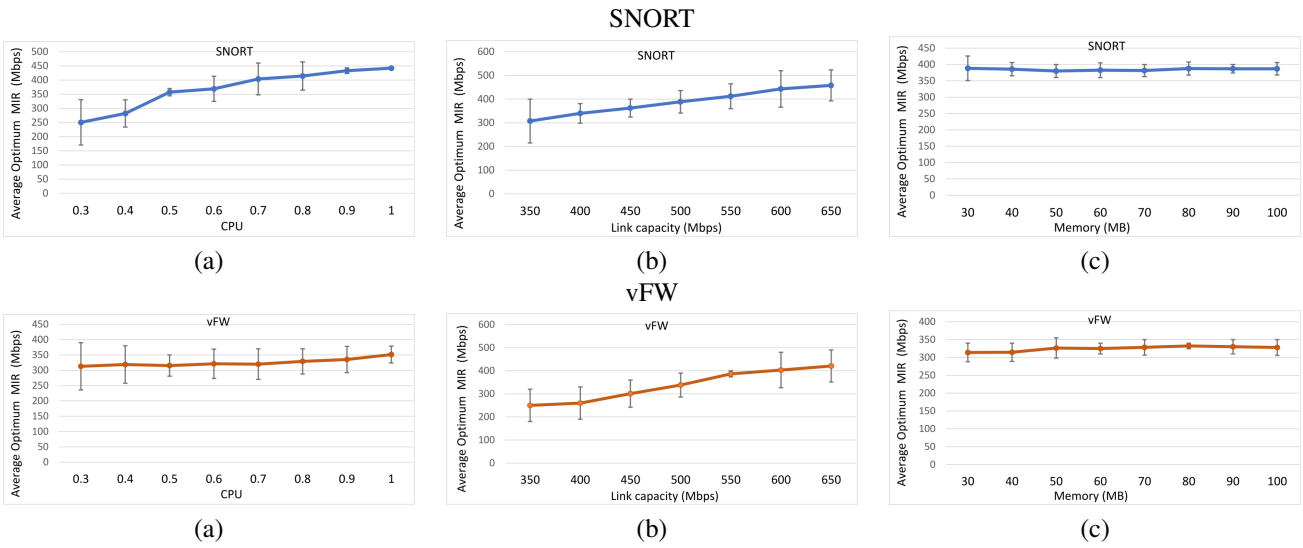


Fig. 7: The Correlation between (a) CPU, (b) Link Capacity, (c) Memory, and the $Optimum\ MIR$ per SNORT and vFW, respectively. Results are annotated with 95% confidence intervals.

MAPE defined in formula (4) measures the deviation from the actual values as a percentage form. Therefore, and unlike MAE and RMSE, it does *not* depend on the data size, hence making it an appropriate auxiliary metric for NAP when facing limited amounts of training or testing input:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \times 100 \tag{4}$$

Finally, *R-squared* (R^2) is given by the following formula:

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}\,. \tag{5}$$

The former formula involves the fraction of the residual sum of squared errors of the regression model (SS_{RES}) over the total sum of squared errors (SS_{TOT}), subtracted from 1, where y_i is the actual value, $\hat{y}_i$ is predicted value and $\bar{y}$ is mean value. Note that the closer the R-square value is to 1, the higher is the correlation to the mean line.

E. Model Parameters Setup & Tuning

1) Predicting the Absolute configuration of Resources: The first role of the models generated by the ***Predictor Manager*** is to predict the absolute configuration of resources that is required for meeting with the given KPIs and $Optimum\ MIR$ in the target environment. In this experiment, the "VNF Performance Datasets" related to the profiled SNORT and vFW VNFs are explored from the Elastic stack and each dataset is divided into a training set and a test set based on a 90%/10% ratio, respectively. It should be pointed out that before starting the training process, the data were normalised by utilising feature scaling. Each model has four input variables under the category of the KPIs and performance metrics and three output variables under the category of the resources. The **input variables** are CPU utilisation, Memory utilisation, Latency, and $Optimum\ MIR$, while the **output variables** are the number of vCPU cores, Memory, and Link Capacity.

The tuning of effective parameters is a prerequisite task for training ML models. Given that the Multi-Layer Perceptron (MLP) with Shallow Neural Network Architecture yielded a low accuracy in our experiments, we proceeded with MLP with Deep Neural Network Architecture, which resulted in higher accuracy. The tuned Smoothing factor parameter for MIMO-GRNN is set as $\sigma = 0.03$ and 0.01, for SNORT, and vFW, respectively, whereas the number of Trees in the Random Forest (RF) model is set to 1200 in both cases. These models are used to predict the absolute configuration of resources. Table IV summarises our MLP parameters. It is worth mentioning that after training the models, predicting the absolute configuration of resources to meet the required KPIs utilising the models generated by the Predictor manager only takes around 44.23, 1.07 and 0.24 second for GRNN-MIMO, MLP and RF, respectively which is very negligible resulting in the possible testing of the models and reconfiguration of resources, reactively.

To compare the models, we have used the error-based evaluation metrics described in section V-D. Table V shows the results of the error metrics along with the R-squared values related to each resource predicted by the models. We found that all three methods provide low error values as well as relatively high R-squared values when we train them with the dataset explored from the Elastic stack. For instance, according to the SNORT, the R-Squared values for MIMO-GRNN are 0.91, 0.97, and 0.74 for vCPU, memory, and link capacity respectively, while MLP gets 0.91, 0.96, and 0.74 and R-Squared values of RF are 0.92, 0.98 and 0.76. Considering only the R-Squared parameter is not sufficient in comparing all investigated models to find the best model to fit our dataset, we have provided various error metrics. As a result, RF appears slightly better than MIMO-GRNN and MLP in terms of accuracy and response time for the prediction of resource configurations for both the SNORT and vFW VNFs. As an example, while predicting the number of vCPU cores needed for the SNORT VNF, the MAPE, MAE and RMSE values for

TABLE V
ERROR-BASED EVALUATION METRICS IN PREDICTING THE ABSOLUTE CONFIGURATION OF RESOURCES

vnf		MIMO-GRNN			MLP			RF		
		CPU	MEM	LC (Mbps)	CPU	MEM	LC (Mbps)	CPU	MEM	LC (Mbps)
SNORT	MAPE (%)	3.21	0.18	5.25	3.83	0.85	5.25	3.16	0.19	4.80
	MAE	0.02	0.10	25.60	0.02	0.56	26.10	0.02	0.08	23.33
	RMSE	0.04	0.79	36.97	0.04	0.99	36.68	0.03	0.68	34.80
	R-squared	0.91	0.97	0.74	0.91	0.96	0.74	0.92	0.98	0.76
vFW	MAPE (%)	4.84	0.67	2.96	5.41	1.91	4.29	3.91	0.92	2.63
	MAE	0.03	0.48	14.76	0.03	1.17	21.41	0.02	0.61	12.90
	RMSE	0.06	2.42	27.33	0.07	2.46	35.73	0.04	2.30	20.03
	R-squared	0.68	0.95	0.81	0.55	0.95	0.65	0.83	0.96	0.90

TABLE IV
PARAMETERS OF THE MLP MODEL TO PREDICT THE ABSOLUTE CONFIGURATION OF RESOURCES

Parameter	SNORT	vFW
Number of neurons in Input Layer	4	4
Number of neurons in Output Layer	3	3
Number of Hidden Layer	55	65
Number of neurons in each Hidden Layer	256	256
Activation Function in hidden layers	selu	selu
Activation Function in the output layer	sigmoid	sigmoid
Epoch	600	600
Batch size	16	16
Optimizer	rmsprop	rmsprop
Learning rate	1e-5	1e-5

TABLE VI
PARAMETERS OF THE MLP MODEL TO PREDICT THE $Optimum\ MIR$

Parameter	SNORT	vFW
Number of neurons in Input Layer	6	6
Number of neurons in Output Layer	1	1
Number of Hidden Layer	3	3
Number of neurons in 1st Hidden Layer	1024	1024
Number of neurons in 2nd Hidden Layer	2048	4096
Number of neurons in 3rd Hidden Layer	4096	4096
Activation Function in hidden layers	relu	relu
Activation Function in the output layer	sigmoid	sigmoid
Epoch	600	600
Batch size	16	16
Optimizer	Adam	Adam
Learning rate	1e-4	1e-5

the RF are 3.16, 0.02, and 0.03, respectively. However, these values are 3.83, 0.02, and 0.04 for the MLP model, and 3.21, 0.02 and 0.04 for the MIMO-GRNN.

2) $Optimum\ MIR$ Recommendation: The second role of models generated by ***Predictor Manager*** is to predict accurately the $Optimum\ MIR$ for resource configurations in which the given KPIs can be met. In this experiment, there are six input variables under the category of the resources and KPIs and performance metrics to predict one output variable. The **input variables** are number of vCPU cores, Memory, Link Capacity, CPU utilisation, Memory utilisation, and Latency, while the **output variable** is $Optimum\ MIR$. Our investigation involves training and test datasets with a ratio of 80:20. For MIMO-GRNN, in the case of SNORT, we set the Smoothing factor parameter to 0.02 ($\sigma = 0.02$), whereas for RF, we set the number of trees to 600. Regarding the vFW, we set $\sigma = 0.005$ for MIMO-GRNN and the number of trees to 1600 for RF. Table VI summarises our MLP parameters. It is noted that after training the models, predicting the *Optimum MIR* utilising the models generated by the Predictor manager only takes around 59.20, 0.70, and 0.53 second for GRNN-MIMO, MLP, and RF, respectively.

Table VII summaries the error-based metrics and R-squared values for predicting the $Optimum\ MIR$ for both profiled VNFs. For Snort VNF, the RF method yields the least error values, even though the other methods provide approximately similar R-squared values. For instance, according to the SNORT, the MAPE, MAE, and RMSE values of RF are equal to 6.55%, 20.50 Mbps, and 30.37 Mbps, respectively. For MIMO-GRNN, on the other, MAPE, MAE, and RMSE are equal to 6.67%, 21.99 Mbps and 32.42 Mbps, respectively. Last, the aforementioned values for MLP are equal to 6.82%, 21.64 Mbps, 31.73 Mbps, respectively. Based on the results, we can conclude that for predicting $Optimum\ MIR$, for SNORT, RF performs better than the rest models (in terms of the accuracy and response time), and for vFW, MIMO-GRNN acts better in terms of accuracy while RF has quite a similar accuracy but with less response time.

TABLE VII
ERROR-BASED EVALUATION METRICS IN PREDICTING THE $Optimum\ MIR$

	SNORT			vFW		
	MIMO-GRNN	MLP	RF	MIMO-GRNN	MLP	RF
MAPE (%)	6.82	6.67	6.55	4.15	14.36	11.12
MAE (Mbps)	21.64	21.99	20.50	11.69	33.03	27.93
RMSE (Mbps)	31.73	32.42	30.37	36.46	46.34	38.86
R-squared	0.89	0.88	0.90	0.84	0.71	0.81

F. The Accuracy of the Performance Prediction

We tested various combinations of resource configurations with the targeted performance metrics and KPIs of 0.95 - 1.1 for CPU utilisation and 2.1 - 3.1 ms for latency. Noted that both CPU utilisation and latency were filtered for SNORT, and latency was filtered for vFW. In this section, we aim to compare the average $Optimum\ MIR$ obtained from the "VNF Performance Dataset" (referred to as "actual") with the predicted results done by the generated model (RF), referred to as "predicted". These tests are divided into 9 filters (F1 to F9), as illustrated in Figure 8, and detailed in Table VIII. In brief, the error rate between the "predicted" $Optimum\ MIR$

TABLE VIII
COMPARING THE OBTAINED $Optimum\ MIR$ UTILISING THE WEIGHTED RESOURCE CONFIGURATION SELECTOR AND THE PREDICTOR MANAGER THROUGH VARIOUS COMBINATIONS OF RESOURCES

Filters	Considered Resources			Metrics & KPIs		SNORT			vFW		
	vCPU cores	Link Capacity (Mbps)	Memory (MB)	CPU Util (%): 95 to 100	Latency (ms): 2.1 to 3.1	Actual Avg. Opt. MIR (Mbps)	Predicted Opt. MIR (Mbps)	Error Rate (%)	Actual Avg. Opt. MIR (Mbps)	Predicted Opt. MIR (Mbps)	Error Rate (%)
F1	0.6 to 0.65	400 to 420	30 to 70			354.0	353.60	0.11	234.03	232.41	0.69
F2	0.6 to 0.65	420 to 450	30 to 70			363.0	359.66	0.9	252.96	251.30	0.66
F3	0.6 to 0.65	450 to 480	30 to 70			370.9	368.29	0.7	286	281.81	1.47
F4	0.65 to 0.68	480 to 500	30 to 70			384.4	386.80	0.6	318.04	323.23	1.63
F5	0.68 to 1	500 to 520	30 to 70			418.6	421.66	0.7	345	339.87	1.49
F6	0.5 to 0.6	500 to 660	30 to 70			342.9	359.15	4.7	358	360.67	0.75
F7	0.65	505	65			389.5	380.02	2.4	330	322.82	2.18
F8	0.65	505	66 to 80			391.4	380.02	2.9	325	315.02	3.07
F9	0.65	505	60 to 65			389.0	381.74	1.8	320.45	325.33	1.52

and the "actual" one in all filters is lower than 5%, which indicates a *highly accurate quality of prediction*.

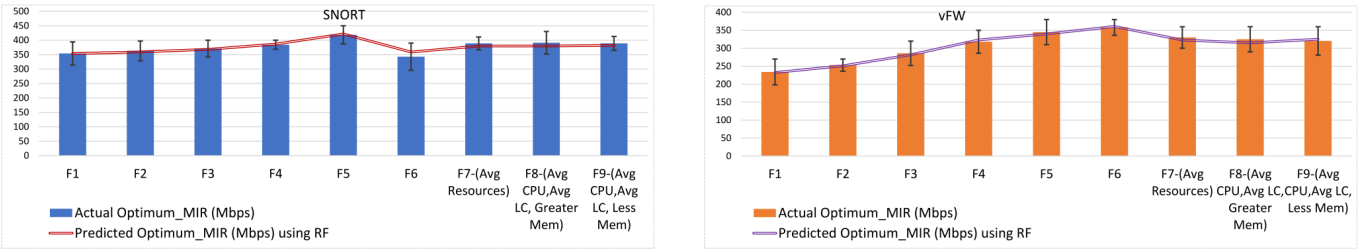
Our results show that both VNFs can handle a higher amount of load ($Optimum\ MIR$) with increasing link capacity, hence the $Optimum\ MIR$ (observed in filters F1 to F3 in Figure 8 and table VIII) is directly proportional to the link capacity. In addition, increasing the amount of vCPU and link capacity simultaneously yields a higher $Optimum\ MIR$ (shown in filters F3 to F5) for both VNFs. It is noticeable that reducing the amount of vCPU and increasing the amount of Link capacity significantly decreases the $Optimum\ MIR$ in SNORT. In comparison, by doing so, the $Optimum\ MIR$ increases in vFW (indicated in filters F5 to F6). We can also realize that in SNORT, the amount of vCPU has more impact on the $Optimum\ MIR$ than the link capacity comparing with the vFW that the link capacity has a greater impact than CPU, which is consistent with the computed weights of resources (W) discussed in section V-B. Straightforwardly, varying the allocated memory does not have a significant impact on the performance metrics of both SNORT and vFW VNFs (see filters F7 to F9).

G. Discussion and Interpretation

The value of performing VNF profiling and adopting profile-based MANO techniques bring a series of advantages when compared to conventional MANO. For example, in a scenario where profiling of VNFs is not performed, the initial deployment of an NS would be done by allocating an arbitrary configuration of the computational and network resources (e.g., by taking an educated guess of a human operator). When this initial configuration is proven not to support the observed demand, the NFVO will have to resort to equally arbitrary policies in order to scale (horizontally or vertically) one or both of the VNFs. Since there is no data-driven or analytical model to guide this process, the time taken by the NFVO to achieve stable configurations might be very long and require multiple iterations before proper convergence. Meanwhile, the quality of service can degrade and trigger undesired effects such as SLA non-compliance and loss of clients. In a scenario where profiling data is available, this convergence time is expected to be very short (indeed, the more precise the model, the quicker convergence should be), which in turn decreases the risk of undesired effects.

Moreover, a service provider might use profiles to choose between similar VNFs from different vendors in order to compose a bigger chain (*day 1 operation*); the NFVO might use profile data to decrease an initially overestimated resource allocation from one VNF in order to prioritize another VNF in the same chain without imposing the need for increasing the overall resource allocation (*day 2 operation*); a service provider might rethink its KPI acceptance criteria and target performance using the profile to estimate the impact of changing them when looking for a way to reduce cost (*ad-hoc management decision*); between other usages. These examples also illustrate how employing profiles help to reduce the dependency on human intervention, trial and error, and bespoke configuration, which is a step forward in the direction of ZSM.

In particular, the case studies presented, although comprised by a single isolated VNF each, can be interpreted as if being part of a hypothetical broader context, which allows us to understand the value of using NAP. We can conceptualise an example based on the 2 VNFs previously presented. Consider a simple NS composed by chaining the vFW and SNORT VNFs in the given sequence, whose only purpose is to protect a set of applications (that expose communication channels via TCP/UDP) against illegitimate access. Because the available profiles clearly point out that the main bottleneck for increasing the supported input rate is the network capacity (the number of vCPUs and amount of memory is modest when compared to commodity datacenter hardware, while the virtual network interface and link capacities will be inversely proportional to the amount of VNFs being hosted inside the same hardware), the NFVO can perform an assertive placement in infrastructural nodes with smaller computational power or memory but good connectivity. Moreover, when fed with the information about the maximum throughput of the network interfaces in the VNFs (which can be given via the software/hardware specification or by an additional profiling experiment), the NFVO is capable of using the profile to decide which service demand/input rate can not be served and trigger an horizontal migration (and associated load-balancing operation). This perspective is valuable to start to unveil the usefulness of the proposed framework in the context of NFV MANO. We have kept the further analysis of diverse resource configurations and varied application and network



(a) (b)

Fig. 8: Comparing Actual and Predicted $Optimum\ MIR$ based on various combinations of resources; (a) SNORT, (b) vFW. Results include 95% confidence intervals.

layer functions out of the scope of this work and have left them as a suggestion for future works in this field.

Finally, it is important to emphasize that for the majority of cases, a service provider will be interested in minimising allocation of CPU cores, memory utilisation and link capacity (or throughput of the network connection), since those factors directly influence operation costs, especially when third-party cloud/edge infrastructure is examined. Therefore it is realistic to assume that most of the profiles should include the related monitoring metrics, similarly to the presented experiments. Additional parameters might be considered depending on the purpose of the NS or specificity of the application layer, e.g., a service for transcoding and video streaming might also require monitoring data about GPU usage and video quality (such as delivered number of frames per second).

VI. Conclusion

We propose a Novel Autonomous Profiling method called *NAP* that can be utilised within the broader context of Zero-touch network & Service Management (ZSM) for the next generation of NFV orchestration. Our *NAP* method integrates three roles: First, a weighted resource configuration selection algorithm, which automatically generates a profiling dataset for the VNFs by selecting the configuration of resources with the highest impact on the performance goals and KPI targets in a limited profiling time. Second, *NAP* creates a model to accurately predict the performance metrics for previously untested resource configurations in which the given performance goals can be met. Last, *NAP* utilises ML-based techniques to estimate the absolute amount of resources required to meet both the given performance goals and the performance metrics in the target environments. The obtained results on *real* datasets related to various type profiled VNFs show that our *NAP* method can predict the untested configuration of resources as well as the performance metrics with significant accuracy. Thus, the model generated by the Predictor Manager, accompanied by our proposed *NAP* method can be used for the next generation of NFV Orchestration during the entire life-cycle management of NSs.

Future work includes extending our current profiling work to cover more resource types. This vision raises the state space of profiling predictions exponentially. We also plan to extend our autonomous profiling method to profile VNFs hosted at edge and cloud environments covering cases such as network slicing and mobility management.

Acknowledgment

This work has received funding from the UK: EPSRC projects INITIATE (EP/P003974/1) and TOUCAN (EP/L020009/1) and from the EU: H2020 projects 5G-VICTORI and MATILDA (grant agreements 857201, 761898).

References

[1] ETSI, "Network transformation: (orchestration, network and service management framework)," https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_White_Paper_Network_Transformation_2019_N32.pdf, Tech. Rep., Oct. 2019, Accessed: 29-05-2020.

[2] ——, "OpenSource MANO - OSM," https://osm.etsi.org/, Accessed: 01-05-2020.

[3] Linux Foundation, "Open Network Automation Platform - ONAP," https://www.onap.org/, Accessed: 01-05-2020.

[4] A. Mestres *et al.*, "A machine learning-based approach for virtual network function modeling," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018, pp. 237–242.

[5] Nfv workload efficiency whitepaper. [Online]. Available: https://tl9000.org/resources/documents/NFVWorkloadEfficiencyWhitepaper.pdf

[6] M. Peuster and H. Karl, "Profile your chains, not functions: Automated network service profiling in devops environments," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2017, pp. 1–6.

[7] ——, "Understand your chains: Towards performance profile-based network service management," in *2016 Fifth European Workshop on Software-Defined Networks (EWSDN)*. IEEE, 2016, pp. 7–12.

[8] L. Cao *et al.*, "Nfv-vital: A framework for characterizing the performance of virtual network functions," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 93–99.

[9] M. Peuster and H. Karl, "Understand your chains and keep your deadlines: Introducing time-constrained profiling for nfv," in *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 240–246.

[10] F. Beye *et al.*, "Towards accurate and scalable performance prediction for automated service design in nfv," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2019, pp. 1–7.

[11] A. Heideker *et al.*, "Profiling service function chaining behavior for nfv orchestration," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 01 020–01 025.

[12] H. Mezaache *et al.*, "Auto-encoder with neural networks for wind speed forecasting," in *2018 International Conference on Communications and Electrical Engineering (ICCEE)*. IEEE, 2018, pp. 1–5.

[13] V. Sciancalepore *et al.*, "z-torch: An automated nfv orchestration and monitoring solution," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1292–1306, 2018.

[14] S. Van Rossem *et al.*, "Monitoring and debugging using an sdk for nfv-powered telecom applications," in *IEEE NFV-SDN2016, the IEEE Conference on Network Function Virtualization and Software Defined Networks*, 2016, pp. 1–5.

[15] ——, "Profile-based resource allocation for virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1374–1388, 2019.

[16] D. Antanasijević *et al.*, "Multiple-input-multiple-output general regression neural networks model for the simultaneous estimation of traffic-related air pollutant emissions," vol. 9, no. 2, pp. 388–397, 2018.

[17] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[18] K. Ogata, *Modern Control Engineering*, 5th ed., ser. Instrumentation and controls series. Prentice Hall, 2010, ch. 2, pp. 17–22.

[19] M. Bunyakitanon *et al.*, "Auto-3p: An autonomous vnf performance prediction & placement framework based on machine learning," *Computer Networks*, vol. 181, p. 107433, 2020.

[20] M. Bunyakitanon, X. Vasilakos, R. Nejabati, and D. Simeonidou, "End-to-End Performance-based Autonomous VNF Placement with adopted Reinforcement Learning," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2020.

[21] Y.-C. Chang, "N-dimension golden section search: Its variants and limitations," in *2009 2nd International Conference on Biomedical Engineering and Informatics*. IEEE, 2009, pp. 1–6.

[22] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[23] M. S. Hossain *et al.*, "A comparative study of vibrational response based impact force localization and quantification using radial basis function network and multilayer perceptron," *Expert Systems with Applications*, vol. 85, pp. 87–98, 2017.

[24] "Snort: SNORT software and documentation," https://www.snort.org/, Accessed: 07-04-2020.

[25] "cAdvisor: Container Advisor," https://github.com/google/cadvisor, Accessed: 07-04-2020.

[26] "Elasticsearch, "Elastic stack"," https://www.elastic.co/elasticsearch/, Accessed: 07-04-2020.

Shadi Moazzeni is a Senior Research Associate with the University of Bristol, Bristol, UK, where she is a member of the Smart Internet Lab and the High Performance Networks research group and the cluster lead researcher of the EU Horizon 2020 5G-VICTORI project. She received her M.Sc. degree from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran in Computer Architecture engineering in 2010 and her Ph.D. in Computer Architecture engineering at the University of Isfahan, Iran in 2018. She was also a Ph.D. visiting researcher at the University of Bologna, Italy, from July 2016 to February 2017. Her current research focuses on the performance and reliability of distributed Software-Defined Networks, Network Function Virtualisation, multi-edge orchestration, monitoring and measuring Performance, and 5G network profiling and orchestration.

Pratchaya Jaisudthi is a lecturer of the Rambhai Barni Rajabhat University, Thailand, and a member of the Smart Internet Lab, currently pursuing his Ph.D. degree in Electrical and Electronic Engineering at the University of Bristol, United Kingdom. His research focuses on ML solutions for Network Function Virtualization. He received his Bachelor and Master degrees in Electrical Engineering from Chulalongkorn University, Thailand in 2006 and 2010 respectively with an emphasis on Telecommunication.

Anderson Bravalheri received his Bachelor and Master degrees in Electrical Engineering (Telecommunications and Telematics) from University of Campinas (UNICAMP), Brazil in 2011 and 2016 respectively. From 2011 to 2017, he was a telecommunication researcher at CPqD working with a variety of technical projects covering from control of optical components and firmware development up to SDN applications for optical networks. Currently, he is a Senior Research Associate at the High Performance Networks Research Group, University of Bristol, UK, and his interests include Optical Communications, Fuzzy Control, Distributed Systems, SDN, NFV and MANO systems

Navdeep Uniyal is a Senior Research Associate at the High Performance Networks Research Group and a member of Smart Internet Lab at the University of Bristol. Previously, he worked at NEC Laboratories Europe, Germany in the Network Research Division where he focused on Information Centric Networks (ICN), NFV Orchestration and Network Microservices. He completed his M.Sc. degree in Distributed Software Systems from Technical University of Darmstadt, Germany in 2015. His research interests lie in the areas of Cloud computing, NFV, SDN, MANO and their applications towards 5G Networks.

Xenofon Vasilakos is a Lecturer at the University of Bristol, the UK. His research is aligned with Bristol Digital Futures Institute (BDFI) and Smart Internet Lab (SIL). Currently, he is the lead researcher of the Zero Downtime Edge Application Mobility (ZeroDEAM) project funded by Samsung Electronics UK. He received the MSc degree in Parallel and Distributed Computer Systems from Vrije Universiteit Amsterdam, and the PhD degree in informatics from the Athens University of Economics and Business with a focus on Information-Centric Networking architectures, protocols, and distributed solutions. He has participated in various EU and national funded research projects such as 5GPPP SliceNet and the FIA award-winning FP7 project PURSUIT. His current research interests include 5G and 6G with a focus on Multi-access Edge Computing based on cognition approaches inspired by machine learning models towards Zero-touch network and Service Management. He is also involved in 5G/6G-related research on the Internet of Things, Software-Defined Networking, Network Function Virtualization, and network Slicing. Dr Vasilakos was a recipient of an excellence fellowship grant from the French government (LABoratoires d'EXcellence), and has received an accolade and awards for his academic performance from the Greek State Scholarship Foundation.

CV: http://pages.cs.aueb.gr/~xvas/pdfs/detailedCV.pdf

Reza Nejabati is currently a full professor and head of High Performance Network Group in the Department of Electrical and Electronic Engineering in University of Bristol. He has established successful and internationally recognized research activities in University of Bristol on "Software Defined Network", "Network Virtualisation" and "Quantum Secured Networking". Under his leadership, these research activities have been transformational and have done major contributions in open, programmable and quantum secured Telecom Infrastructure. His research received the prestigious IEEE Charles Kao Award 2016. Building on his research portfolio over the last 3 years, he co-founded a successful start-up company (Zeetta Networks Ltd).

Dimitra Simeonidou is a Full Professor at the University of Bristol, the Co-Director of the Bristol Digital Futures Institute, and the Director of the Smart Internet Lab. Her research is focusing in the fields of high performance networks, programmable networks, wireless-optical convergence, 5G/B5G and smart city infrastructures. She is increasingly working with Social Sciences on topics of digital transformation for society and businesses. Dimitra has been the Technical Architect and the CTO of the smart city project Bristol Is Open. She is currently leading the Bristol City/Region 5G urban pilots. She is the author and co-author of over 500 publications, numerous patents and several major contributions to standards. She has been co-founder of two spin-out companies, the latest being the University of Bristol VC funded spin-out Zeetta Networks, http://www.zeetta.com, delivering SDN solutions for enterprise and emergency networks. Dimitra is a Fellow of the Royal Academy of Engineering, a Fellow of the Institute of Electrical & Electronic Engineers, and a Royal Society Wolfson Scholar.